Journal of Big Data Analytics in Transportation (2020) 2:223–233
https://doi.org/10.1007/s42421-020-00027-8

ORIGINAL PAPER

# Technical and Economic Feasibility Assessment of a Cloud-Enabled Traffic Video Analysis Framework

**Tongge Huang**[1] · **Anuj Sharma**[1]

Received: 20 July 2020 / Revised: 29 October 2020 / Accepted: 22 November 2020 / Published online: 22 December 2020


## Abstract

Computer vision techniques are expected to significantly improve the development of intelligent transportation systems (ITS), which are anticipated to be a key component of future Smart City frameworks. Powered by computer vision techniques, the conversion of existing traffic cameras into connected "smart sensors" called intelligent video analysis (IVA) systems has shown the great capability of producing insightful data to support ITS applications. However, developing such IVA systems for large-scale, real-time application deserves further study, as the current research efforts are focused more on model effectiveness instead of model efficiency. Therefore, we have introduced a real-time, large-scale, cloud-enabled traffic video analysis framework using the NVIDIA DeepStream and NVIDIA Metropolis. In this study, we have evaluated the technical and economic feasibility of our proposed framework to help traffic agency to build IVA systems more efficiently. Our study shows that the daily operating cost for our proposed framework on Google Cloud Platform is less than $0.14 per camera, and that, compared with manual inspections, proposed framework achieves an average vehicle-counting accuracy of 83.7% on sunny days.

**Keywords** ITS · Real-time traffic video analysis · Cloud-enabled framework · Economic assessment · Large-scale feasibility · Vehicle counting

## Introduction

The Smart City (SC) technology that makes our daily lives to be more connected and informed is growing rapidly throughout the world. It represents a new framework which integrates information and communication technology (ICT) and Internet-of-Things (IoT) technology to improve citizens' quality of life (Memos et al. 2018). Nowadays, many organizations and corporations have proposed innovative solutions to help the development of the Smart City, such as NVIDIA's Metropolis, Siemens' MindSphere, and Huawei's OceanConnect. As the crucial component of Smart City technology, the IoT-driven and cloud-enabled intelligent transportation systems (ITS) have drawn large amounts of attention to make driving safer, greener and smarter. The ITS market is likely to grow to approximately $130 billion by 2025 (Mishra et al. 2019). According to the USDOT's research-based initiative named "Beyond Traffic 2045: The Smart City Challenge", the department has expanded the resources pool by more than $500 million to support innovative Smart City projects, so as to improve the performance of the transportation system by addressing the issues of congestion, safety, etc. (Gandy and Nemorin 2020).

In traditional ITS, the loop detector and microwave sensors have been the primary choices to gather traffic information. However, the data collected by these traditional sensors are aggregated on the sensor side so that detailed information is lost, and installing such sensors is also costly (Dai et al. 2019). In addition, traditional video surveillance requires watchstanders to pay close attention to hundreds of screens simultaneously, which is challenging and tedious (Liu et al. 2013). Yet there exists already a massive number of traffic surveillance cameras widely installed on the road network across the US to ensure traffic safety. Thus, converting the existing traffic surveillance cameras into connected "smart sensors", also called intelligent video analysis (IVA), has gained a large amount of attention over the past several years.

✉ Tongge Huang
tongge@iastate.edu

[1] Civil, Construction, and Environmental Engineering Department, Iowa State University, Ames, IA 50011, USA

Typically, based on computer vision (CV) techniques, traffic cameras have been considered capable sensors for data extraction and analysis in various ITS applications, such as congestion detection, vehicle counting, and traffic anomaly detection (Chakraborty et al. 2018; Dai et al. 2019; Kumaran et al. 2019). Such ITS approaches mainly consist of vehicle detection and tracking. For vehicle detection, deep learning-based methods have been proposed due to the development of convolutional neural networks (CNN) such as the two-stage methods like Fast R-CNN (Girshick 2015), Faster R-CNN (Ren et al. 2015) and mask R-CNN (He et al. 2017) and one-stage methods like YOLOv3 (Redmon and Farhadi 2018), SSD (Liu et al. 2016), and RetinaNet (Lin et al. 2017), etc. For vehicle tracking, many tracking algorithms have been proposed such as DeepSORT (Wojke et al. 2017), IOU (Bochinski et al. 2018), KLT (Lucas et al. 1981) and DCF (Lukezic et al. 2017). Taking advantage of computer vision techniques, Dai et al. (2019) have proposed a video-based vehicle-counting framework using YOLOv3 and kernelized correlation filters (KCF) as detector and tracker, respectively, which achieved an 87.6% counting accuracy running at 20.7 fps, and their video real-time rate was 1.3 (by taking the duration of their framework's runtime divided by the duration of their test videos). Also, Meng et al. (2020) have proposed a correlation-matched tracking algorithm combined with SSD for vehicle counting which reached 93% counting accuracy running at 25 fps on an expressway dataset.

However, current research efforts on deep learning-based approaches have been more focused on model effectiveness instead of efficiency (Liu et al. 2018). Traditional analysis frameworks need to store and process video streams locally, which is unfeasible due to intensive computation cost and processing latency, especially for large-scale camera systems. Such offline approaches, with their high latency, lead to delayed decisions, which is not tolerable for IoT-driven ITS applications like accident detection and congestion reduction (Ferdowsi et al. 2019). Hence, by pushing the deep learning techniques close to their data sources, the edge computing frameworks show potential ability for real-time intelligent video analysis (Liu et al. 2018). To achieve this purpose, NVIDIA has developed the AI-powered high-throughput and scalable inference framework called NVIDIA DeepStream, which has enabled edge computing techniques for multi-GPU and multi-stream video analysis (NVIDIA Deepstream 2020). In addition, NVIDIA Metropolis (NVIDIA Metropolis 2019), as an end-to-end video analysis platform, has been developed to explore the valuable data from the AI and IoT devices for frictionless retail, streamlined inventory management and so on.

Designing the next-generation ITS applications is required to efficiently support the development of Smart Cities. Thus, the cloud server is playing an important role in integrating the edge computing node or IoT devices horizontally, to provide real-time and dynamic message exchanges for supporting modern ITS applications (Iqbal et al. 2018; Petrolo et al. 2017). As summarized in Eltoweissy et al. (2019), running ITS applications in the cloud has the following benefits. First, cloud servers give users a massive pool of computation resources that eliminate the need to install and maintain private clusters. Second, cloud servers provide flexible charging plans that allow users to pay for resources on a short-term basis. In addition, a properly configured cloud server provides worldwide remote access that helps traffic agency, stakeholders, and governmental organizations access and understand their data easily. Therefore, edge-computing-enabled real-time video analysis frameworks deserve further study to support large-scale Smart-City-based ITS applications. Further, cost estimation for operating such cloud-enabled IVA frameworks is needed to help the traffic agency build the system efficiently.

In this study, we introduce a real-time, large-scale, cloud-enabled traffic video analysis framework using NVIDIA DeepStream and NVIDIA Metropolis, which are recognized as the cutting-edge AI-powered video analysis toolkit and end to end IVA platform. For this study, hundreds of live video streams recorded by traffic CCTV cameras across Iowa were decoded and inferred using NVIDIA DeepStream. The model inference results generated by NVIDIA DeepStream were processed and analyzed using the big data processing platforms Apache Kafka (Kafka 2019), Apache Spark (Spark 2020), Elasticsearch and Kibana (Elastic 2020) for data transmitting, processing, indexing, and visualizing, respectively. Toward the goal of efficient access and standardized deployment, each component of the framework is containerized using Docker (Docker 2020). Taking advantage of the benefits of cloud servers, we have deployed the proposed framework on Google Cloud Platform (GCP) and estimated the operating costs in detail from associated billing reports. Then we evaluated the technical feasibility of our proposed framework by measuring its vehicle-counting accuracy. Further, we present the extensibility of our proposed framework with the discussion of its limitations for supporting future real-time ITS applications.

*Contributions* The main contributions of this study are (1) We introduce a real-time, large-scale, cloud-enabled traffic video analysis framework using the cutting-edge video analysis toolkit and end to end platform; (2) we present detailed computational resource usage and operating costs to help traffic agency develop such IVA frameworks; and (3) we evaluate the technical feasibility of our proposed framework for real-time, large-scale implementation by discussing its vehicle-counting accuracy under various scenarios.

*Outline* The remainder of this paper is organized as follows: the next section provides the background and materials for our proposed IVA framework in detail. The subsequent

section discusses the capability and efficiency of our proposed framework, followed by the economic assessment of operating the framework on GCP. Also, the technical feasibility of applying this framework is discussed. Finally, we discuss the potential abilities, limitations and scope of future work on the proposed framework.

## Background and Materials

In this section, we present the background and materials we have used in this study. Following the recent AI-based IoT projects, especially NVIDIA smart parking (NVIDIA IoT 2019), NVIDIA real-time video redaction (Shah et al. 2020), and real-time analysis of popular Uber locations (Carol 2018), we have introduced an end to end practical cloud-enabled traffic video analysis framework for real-time, large-scale traffic recognition. More specific, proposed framework is based on the NVIDIA Metropolis design, especially the NVIDIA smart parking application. Proposed framework consists of two modules, namely a perception module and an analysis module, with end-to-end implementation on GCP as discussed below.

### Perception Module

As mentioned in "Introduction", NVIDIA DeepStream is an accelerated AI-powered framework based on GStreamer, which is a multimedia processing framework (Gstreamer 2019). The key features of NVIDIA DeepStream can be summarized as follows. First, it provides a multi-stream processing framework with low latency to help developers to build IVA frameworks easily and efficiently. Second, it delivers high-throughput model inference for object detection, tracking, etc. Also, it enables the transmission and integration between model inference results and the IoT interface such as Kafka, MQTT or AMQP. Combined with the NVIDIA Transfer Learning Toolkit (TLT), NVIDIA DeepStream provides highly flexible customization for developers to train and optimize their desired AI models.

Therefore, NVIDIA DeepStream is used as our perception module as shown in Fig. 1. The main components of perception module are (1) video pre-processing, which decodes and forms the batches of frames from multiple video stream sources; (2) model inference, which loads TensorRT-optimized models for vehicle detection; (3) OpenCV-based tracker, which tracks the detected vehicles to provide detailed trajectories; (4) model inference results like bounding boxes and vehicle IDs drawn on composite frames for visualization in on-screen display; and (5) model inference results converted and transmitted through the IoT interface by the message handler using the JSON format.

The perception module supports various video stream inputs like the Real-Time Streaming Protocol (RTSP), recorded video files and V4L2 cameras. In this study, we used live video streams recorded by traffic cameras through the public RTSP managed by the Iowa Department of Transportation. For the primary vehicle detector, we used TrafficCamNet (NVIDIA TLT 2020), which utilizes ResNet18 as its backbone for feature extraction (He et al. 2016). The model was trained to detect four object classes, namely car, person, roadsign, and two-wheeler. A total of 200,000 images, which includes 2.8 million car class objects captured by traffic signal cameras and dashcams, were involved in the training. The model's accuracy and F1 score, which were measured against 19,000 images across various scenes, achieved 83.9% and 91.28%, respectively (NVIDIA TLT 2020).

For the multi-object tracking (MOT) task, occlusion and shadow are inevitable issues, especially for vehicle tracking in complex scenarios. Thus, to recognize traffic patterns more robustly and efficiently, we used the NvDCF tracker, which is a low-level tracker based on the discriminative correlation filter (DCF) and Hungarian algorithm. We recommend interested readers refer to NVIDIA Deepstream (2020) for more details.

### Analysis Module

As mentioned above, the model's inference results are delivered to the data transmission interface. Here, the analysis module processes and analyzes the resulting metadata for further ITS applications as shown in Fig. 2.

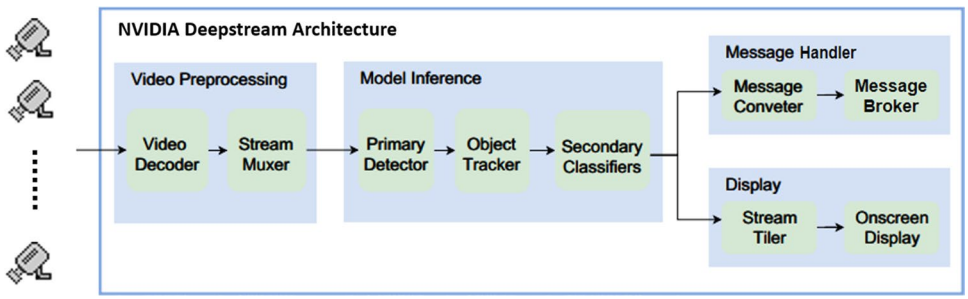


**Fig. 1** Perception module architecture

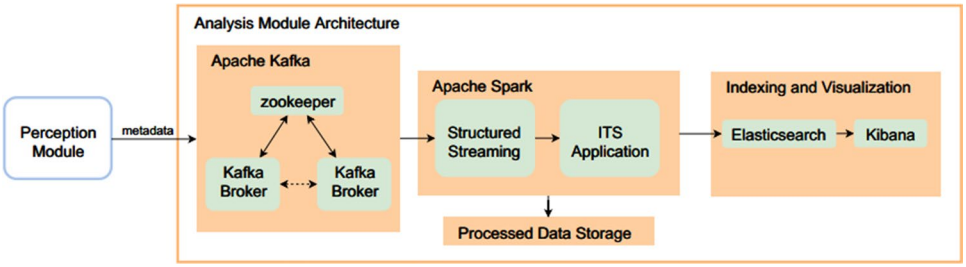


**Fig. 2** Analysis module architecture

First, we used Apache Kafka to handle the data transmissions on a large-scale basis. Kafka has been introduced as a real-time, fault-tolerant distributed streaming data platform. It takes input data from various sources such as mobile devices, physical sensors, and web servers. The data collected from edge devices (say traffic cameras) are published to the Kafka broker with predefined Kafka topics. Each Kafka topic is maintained in many partitions to achieve the purpose of high throughput and fault tolerance. In addition, data from multiple sources can be connected to different Kafka brokers which are managed by a zookeeper.

For large-scale batch data processing in streaming environments, Apache Spark was used in this study. Spark is recognized as a computationally efficient, large-scale data processing platform. Using Resilient Distributed Datasets (RDD) and Directed Acyclic Graphs (DAG), Spark provides a high-speed and robust fault-tolerant engine for big data processing. In analysis module, Spark acts like a "consumer," subscribing to the streaming data from Kafka for further data cleaning, data analysis, trend predictions with machine learning, etc. More specifically, Spark structured streaming is used, which is an exactly-once operation built on the Spin a unbounded table which allows the append operation for "micro-batch" processing.

### Cloud-Enabled Implementation

As mentioned in "Introduction", deploying a large-scale IVA framework on a cloud server is flexible and powerful as a cloud server provides easy accessibility for traffic agency to maintain and manage IVA systems remotely. In this study, we have implemented our proposed framework on the Google Cloud Platform (GCP), which is a scalable and powerful cloud computing service. Also, Docker (version 19.03) has been used to deploy the proposed framework efficiently. Each component mentioned in "Perception module" and "Analysis module" has been containerized and is managed by Docker and Docker Compose. The entire proposed framework is shown in Fig. 3.

In summary, the perception Docker takes the live video streams from multiple cameras as inputs through the RTSP. Then, the decoded video frames are inferred by the trained AI model, and the model inference results are sent to the analysis Docker for further data processing and visualization. The entire framework is operated in GCP, which can be remotely accessed by local machines for the purposes of maintenance, inspection, diagnosis, etc.

## Case Studies and Discussion

In this study, there were 160 traffic surveillance cameras involved. We accessed their live video streams using the public RTSP managed by the Iowa Department of Transportation (Iowa DOT). These live videos are encoded following the H.264 standard with the resolution of $480 \times 270$. In the following sections, we present in detail the resources usage, operating costs and technical feasibility of proposed framework running on GCP.

### Economic Assessment

The GCP instance we used belongs to the N1 series powered by the Intel Skylake CPU platform. We used 8 virtual CPU (vCPU) cores with 30 GB memory and Ubuntu 18.04 LTS operating systems for evaluation. In addition, we used 2 NVIDIA T4 GPUs for video analysis and a 3-TB persistent disk (PD) for data storage.

As mentioned in "Perception module", the perception module can take multiple video streams as inputs, so it requires developers to twiddle the number of video streams (traffic cameras) they need to process simultaneously in each single GPU so that they achieve a balance between performance and cost. We used built-in NVIDIA DeepStream functionality to determine the processing latency of three main components with various numbers of traffic cameras. Results of this latency analysis are presented in Fig. 4.

For the purpose of processing data in near real time, we distributed 80 cameras to each T4 GPU so that the overall processing latency caused by the perception module was less than 1 s. Thus, the total of 160 cameras was split into two T4 GPUs where the average GPU memory usage was around 65%. In other words, the number of cameras could be decided by a traffic agency based on their tolerance of processing delay, without necessarily exceeding the total GPU memory.

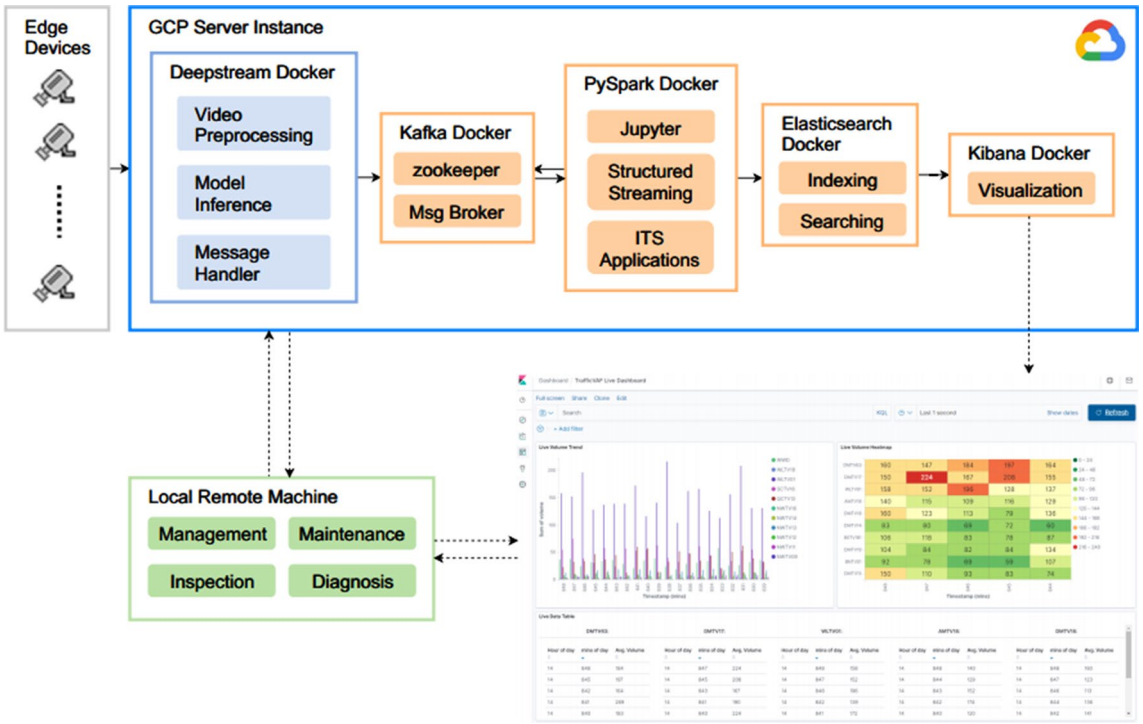


**Fig. 3** Overall intelligent video analysis (IVA) framework

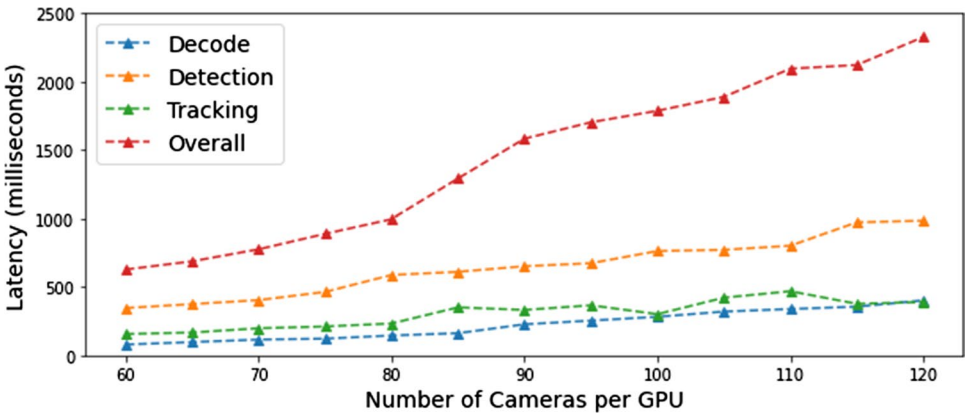


**Fig. 4** Latency for the perception module's main components when simultaneously running different numbers of cameras per GPU

In the perception module, there are six consecutive frames skipped at the stage of model inference for computational efficiency. The coexisting CUDA 10.2 and CUDA 10.1, GPU driver version 418+, and TensorRT 7.0 were used to improve the performance of the NVIDIA T4 GPU according to the T4 workaround specification (NVIDIA Deepstream 2020), and Kafka 2.4, Spark 2.4.5, and Elasticsearch/Kibana 7.5.2 were used in the analysis module. In Table 1, we summarize the computational resources usage for running our framework on GCP.

To measure the operating costs in detail, we implemented the proposed end to end framework in GCP for 5 days. The GCP provides a sustained use discount based on monthly

**Table 1** Summary of computational resources usage on GCP

| Container | CPU (%) | Memory (%) | GPU memory (per GPU) (%) | GPU util (per GPU) (%) |
|---|---|---|---|---|
| Perception0 | 22.3 | 6.9 | 65.5 | 71.0 |
| Perception1 | 21.6 | 6.5 | 64.1 | 69.0 |
| Kafka | 3.8 | 2.2 | 0 | 0 |
| Spark | 30.6 | 29.8 | 0 | 0 |
| Elasticsearch | 10.3 | 3.3 | 0 | 0 |
| Kibana | 0.18 | 0.4 | 0 | 0 |

usage level. In Fig. 5a and b, we show the average daily cost with and without the sustained use discount. The daily operating cost for 160 cameras was $21.59 with the sustained use discount (assuming such a traffic video analysis framework would be operating for the long term). The GPUs and vCPUs are the two major costs, which, respectively, took 46.4% and 19.7% of the total cost, while the persistent storage, network, and RAM together took 33.9% of the total cost. Thus, the daily cost of proposed framework for each camera was $0.135, leading to the yearly cost per camera of $49.30. In Iowa, there are 390 operating traffic surveillance cameras on the road network. Thus, turning all these surveillance cameras into connected "smart sensors" would cost $1601.40 per month using such a cloud-enabled system, with the benefit of eliminating additional infrastructure costs.

## Feasibility Assessment

In this section, we present the feasibility of our proposed framework by measuring its vehicle counting accuracy compared with manual inspection and counting via nearby radar sensors. There were 12 cameras with different viewing angles selected for this evaluation, as shown in Fig. 6. The selected cameras are representative since they cover the most common camera viewing angles across Iowa.

For each camera, we manually counted the vehicles that passed a predefined region of interest (ROI) for 5 min on a sunny day and 4 min on a rainy/cloudy day around 11 am as the ground truth, thus there were 108 min of video involved in our evaluation. The ROI for these cameras was set from 90 to 260 pixels counting from the bottom left along the *y*-axis. For instance, the ROIs for camera 1 and camera 2 were from 10 to 90 pixels and from 10 to 200 pixels along the *y*-axis, respectively. The accuracy of measurement of our vehicle counting application was defined as

$$Cr = \frac{C_g - |C_g - C_e|}{C_g}, \tag{1}$$

where $C_r$, $C_g$, and $C_e$ denote the (1) correct counting rate, (2) counting ground truth, and (3) counting estimated by proposed framework. The results of this evaluation of our vehicle counting application for different cameras under both sunny and cloudy/rainy environments are shown in Table 2.

In addition, we show trajectory plots for two sample cameras under the sunny and rainy day scenarios in Fig. 7. For sunny days, the results show that the proposed framework is working as expected in that its average counting accuracy achieved 90.1%, except with the cameras 3 and 12. During the test periods, camera 3 was suffering from the network lag issue, so the video frames transmitted to the perception module were nonconsecutive. Such a lag issue will make the tracking model produce labels repeatedly for the same vehicle. For camera 12, the perception module barely captured the passing vehicles, which indicates that the model needs to be fine-tuned to fit this scenario involving distant viewing angles. In addition, the results show that the counting accuracy was affected on rainy days for camera 6, camera 9, and camera 10 which, as shown in Fig. 7d, were exposed to the rain.

To further measure the feasibility of our proposed framework, we compared the daily volume pattern captured by our testing cameras versus by their nearby microwave radar sensors, as shown in Fig. 8. However, it is worthy of mention that matching the detection areas of camera sensors and radar sensors point by point is hard since our camera sensors were zoomed into the ROI with ramps, interchanges and minor roads included, as discussed in "Feasibility assessment". Therefore, the overall traffic pattern comparison is more meaningful in showing the agreement between these two types of sensors.

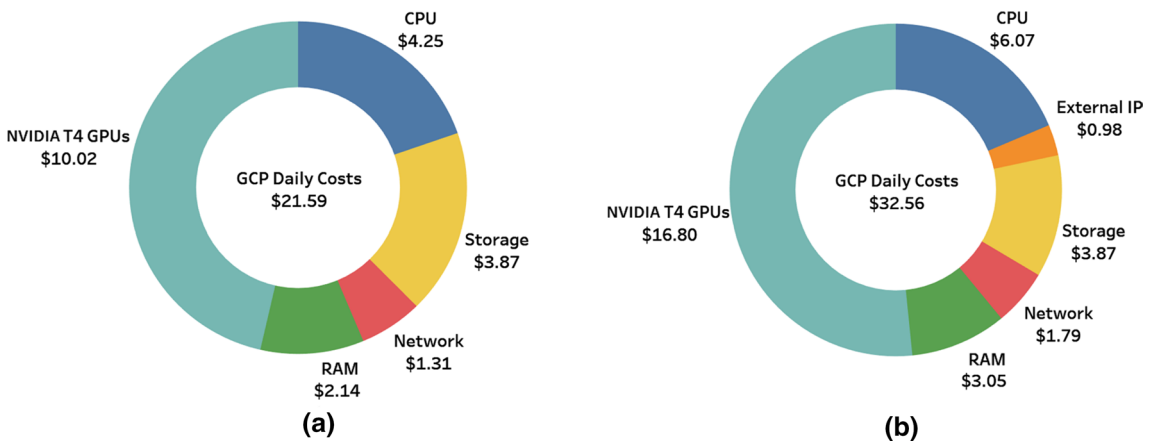


**Fig. 5** Daily operating cost for 160 cameras **a** with sustained use discount and **b** without sustained use discount


**Fig. 6** Sampled cameras with different viewing angles

**Table 2** Experimental results for different cameras in different environments

| Scenes | Sunny | | | | Rainy/Cloudy | | | |
|---|---|---|---|---|---|---|---|---|
| | $C_g$ | $C_e$ | $\lvert C_g - C_e \rvert$ | $C_r$ (%) | $C_g$ | $C_e$ | $\lvert C_g - C_e \rvert$ | $C_r$ (%) |
| Cam 1 | 511 | 539 | 28 | 94.5 | 344 | 401 | 57 | 83.4 |
| Cam 2 | 108 | 103 | 5 | 95.4 | 92 | 98 | 6 | 93.5 |
| Cam 3 | 381 | 470 | 89 | 76.6 | 268 | 391 | 123 | 54.1 |
| Cam 4 | 326 | 336 | 10 | 96.9 | 223 | 221 | 2 | 99.1 |
| Cam 5 | 129 | 148 | 19 | 85.3 | 97 | 102 | 5 | 94.8 |
| Cam 6 | 126 | 138 | 12 | 90.5 | 91 | 113 | 22 | 75.8 |
| Cam 7 | 107 | 113 | 6 | 94.4 | 101 | 104 | 3 | 97.0 |
| Cam 8 | 65 | 56 | 9 | 86.2 | 44 | 38 | 6 | 86.4 |
| Cam 9 | 179 | 211 | 32 | 82.1 | 90 | 292 | 202 | −124.4 |
| Cam 10 | 106 | 112 | 6 | 94.3 | 73 | 31 | 42 | 42.5 |
| Cam 11 | 359 | 291 | 68 | 81.1 | 259 | 234 | 25 | 90.3 |
| Cam 12 | 51 | 14 | 37 | 27.5 | 31 | 10 | 21 | 32.3 |

The data of the daily volume plots were aggregated in 10-min intervals. For camera 1 and camera 11, the volume trend after 3 pm is not presented, since these cameras turn the opposite direction after 3 pm, which is beyond the scope of this study. Also, the comparison of camera 2, camera 5 and camera 7 was impacted by the nearby interchanges and ramps, especially for camera 7 whose closest radar sensor had failed (thus we had to use the radar sensor located two interchanges above instead). In addition, for camera 11, the vehicles on a minor road which was far from the camera missed detection. Similarly, as described earlier, our model only occasionally captured the passing vehicles within distant view of camera 12. In Tables 3 and 4, we aggregate the data in 3-h intervals and present the differences using the absolute error (AE) and relative error (RE) defined as:

$$AE = |V_{\text{camera}} - V_{\text{radar}}| \tag{2}$$

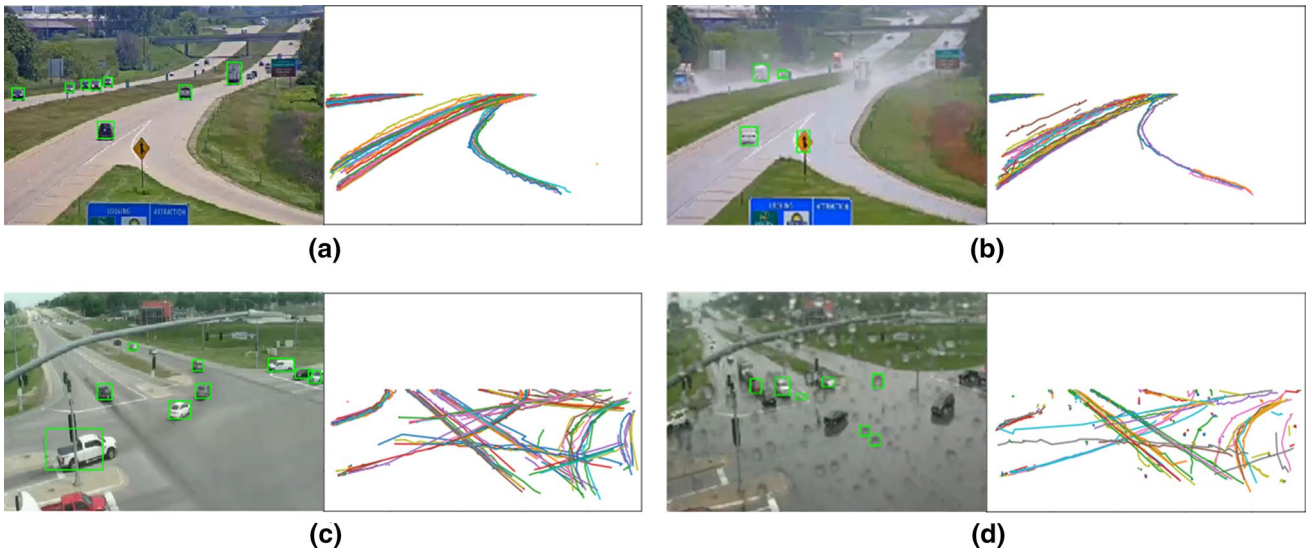


**Fig. 7** Camera views and trajectory plots on sunny and rainy days: **a** Camera 5 sunny day, **b** Camera 5 rainy day, **c** Camera 9 sunny day, **d** Camera 9 rainy day

$$RE = \frac{|V_{\text{camera}} - V_{\text{radar}}|}{V_{\text{radar}}} \tag{3}$$

where $V$ represents the 3-h aggregated traffic volumes.

Overall, the traffic patterns captured by the testing cameras using computer vision techniques versus by their nearby radar sensors are comparable, except with camera 12 where, as noted earlier, model fine-tuning for distant viewing angles is needed. Also, the results shown in both Fig. 8 and Table 4 demonstrate that the model does not perform well during the night (between 9 pm and 6 am), which indicates the model also requires fine-tuning for night-time applications.

As mentioned above, one advantage of our proposed framework is its large-scale production of vehicle trajectories on a real-time basis, which has potential for use in ITS applications like anomaly detection, speed violation detection, wrong-way detection , etc. In Fig. 9, we present the extensibility of our proposed framework by showing three anomalous events on a freeway with their corresponding image coordinates' moving speed calculated from the associated vehicle trajectories. The mean and minimum image coordinates' moving speed was calculated from all the detected vehicles and their trajectories during a certain time period (say 1 second). It is clearly observable that, by applying appropriate thresholds or advanced algorithms, such real-time trajectories generated by our proposed framework could help traffic agency to quickly address anomalous events.

For long-term, real-world operations, it should be noted that the current GCP does not support live migration for cloud instances with GPUs during the host maintenance period, which, as tested in our experiments, typically occurs once a week. In other words, the running programs will be disrupted automatically by the GCP system for the purpose of host maintenance. Therefore, to minimize the disruption of their programs, users need to prepare for their workload's transition through this system restart by monitoring the maintenance schedule, which is usually announced 1 h a head of system termination.

## Conclusions and Future Work

In this study, we have introduced an real-time, large-scale, cloud-enabled traffic video analysis framework using NVIDIA DeepStream and NVIDIA Metropolis, and we have evaluated the proposed framework from two perspectives: economics and feasibility. The proposed framework consists mainly of a perception module and analysis module. The perception module takes multiple live video streams as inputs, and outputs insightful inferred metadata which are produced by the embedded AI model. Then these resulting metadata are transmitted to the analysis module through Kafka. In the analysis module, Spark consumes the data and forms a dynamic unbounded table for batch processing. Finally, the processed data are indexed and visualized by Elasticsearch and Kibana.

Our study demonstrates the proposed framework is both economically efficient and technically feasible. From the perspective of economics, the results show that the daily operating cost for each camera is less than \$0.14, so the yearly operating cost per camera is less than \$50. In addition, we have presented the processing latency of our framework's main components and its usage of cloud computational

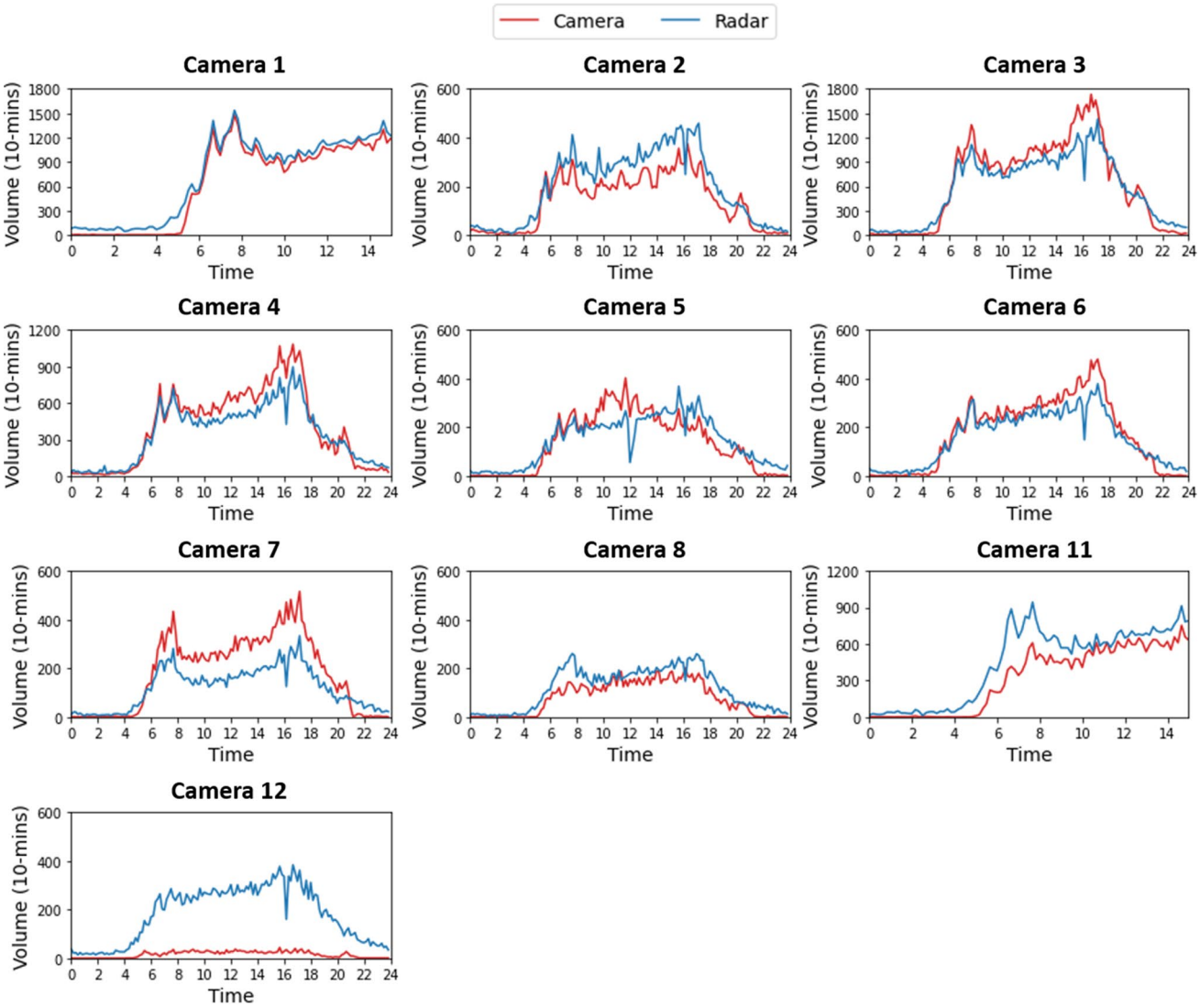


**Fig. 8** Daily traffic volume data (10-min agg) captured by cameras and nearby radar sensors on Monday, June 1st

**Table 3** Absolute error (AE) of daily volume between cameras and radar sensors

| Scenes | Time of day | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 0–3 | 3–6 | 6–9 | 9–12 | 12–15 | 15–18 | 18–21 | 21–24 |
| Cam 1 | 1276 | 2413 | 1277 | 1397 | 1394 | 38 | NA | NA |
| Cam 2 | 220 | 313 | 1266 | 1406 | 1860 | 1946 | 619 | 359 |
| Cam 3 | 619 | 1135 | 2417 | 1690 | 2772 | 5520 | 496 | 1814 |
| Cam 4 | 331 | 218 | 678 | 1848 | 2672 | 3701 | 319 | 877 |
| Cam 5 | 241 | 480 | 270 | 1622 | 554 | 1168 | 781 | 735.5 |
| Cam 6 | 264 | 267 | 341 | 499 | 795 | 1776 | 434 | 597 |
| Cam 7 | 176 | 240 | 1653 | 1829 | 2285 | 2964 | 1265 | 685 |
| Cam 8 | 151 | 450 | 1405 | 474 | 661 | 1104 | 698 | 569 |
| Cam 11 | 553 | 1914 | 5051 | 1573 | 1868 | 160 | NA | NA |
| Cam 12 | 357 | 1048 | 3857 | 4282 | 4678 | 5206 | 3002 | 1150 |

**Table 4** Relative error (RE) of daily volume between cameras and radar sensors

| Scenes | Time of Day | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 0–3 | 3–6 | 6–9 | 9–12 | 12–15 | 15–18 | 18–21 | 21–24 |
| Cam 1 | 95.9 | 60.3 | 6.2 | 7.8 | 6.5 | 3.1 | NA | NA |
| Cam 2 | 52.0 | 23.4 | 24.4 | 27.9 | 30.2 | 27.9 | 22.5 | 59.0 |
| Cam 3 | 78.0 | 43.8 | 16.2 | 12.1 | 16.7 | 27.1 | 4.5 | 59.8 |
| Cam 4 | 49.5 | 11.4 | 7.3 | 22.8 | 27.9 | 30.0 | 5.4 | 41.1 |
| Cam 5 | 90.9 | 53.9 | 8.0 | 42.6 | 13.6 | 24.1 | 29.6 | 77.5 |
| Cam 6 | 83.9 | 29.4 | 9.0 | 12.2 | 16.8 | 33.5 | 16.3 | 64.7 |
| Cam 7 | 86.8 | 33.3 | 47.8 | 69.3 | 69.7 | 69.4 | 61.5 | 84.8 |
| Cam 8 | 91.0 | 65.4 | 40.6 | 16.6 | 19.2 | 27.8 | 38.4 | 89.0 |
| Cam 11 | 89.8 | 74.2 | 40.1 | 14.5 | 14.4 | 20.2 | NA | NA |
| Cam 12 | 100.0 | 89.8 | 91.0 | 90.0 | 90.6 | 91.0 | 93.3 | 96.5 |

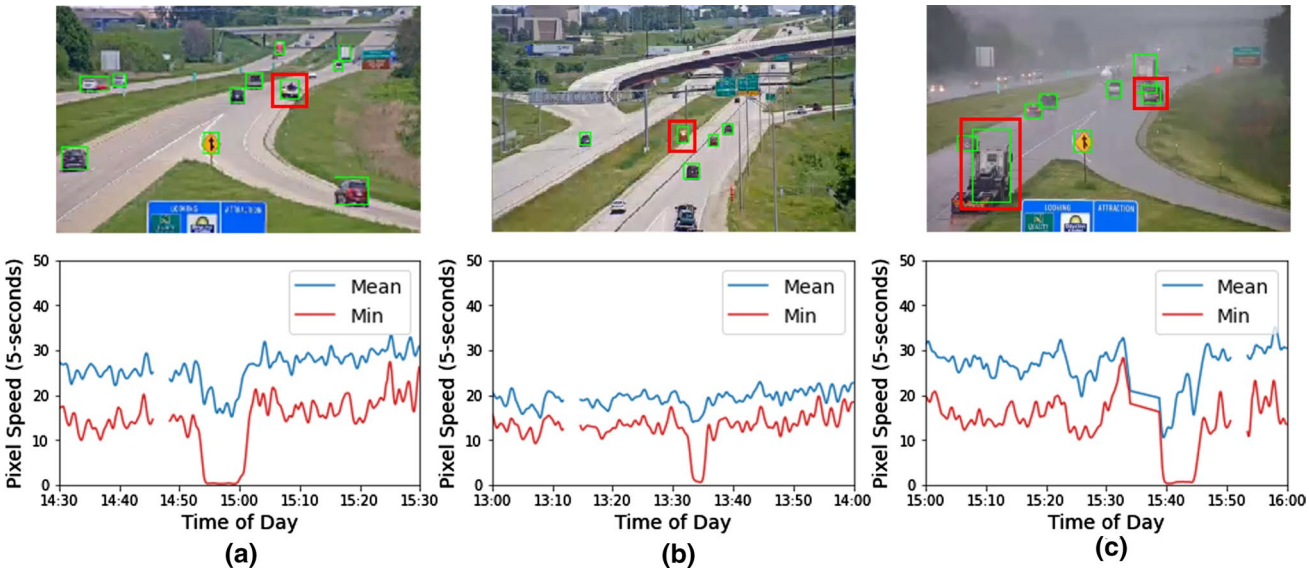


**Fig. 9** Anomalous events with image coordinates' moving speed

resources, which can help developers design AI-powered traffic video analysis frameworks accordingly based on their requirements. From the perspective of technical feasibility, we have measured the accuracy of a vehicle-counting application using the live video streams recorded by 12 cameras with different viewing angles. The results show that for most of the tested viewing angles, the proposed framework is able to produce the expected vehicle-counting results compared with both manual inspections and the count from nearby microwave sensors. Further, we have demonstrated the potential ability of our proposed framework for future real-time ITS applications by showing sample anomalous events with image coordinates' moving speed calculated in seconds.

It should be noted that such real-time ITS applications will likely always suffer from the issue of network lag. As discussed in "Case studies and discussion", the nonconsecutive frames caused by network lag affect the performance of our proposed framework for both vehicle detection and tracking modeling. Thus, developers need to work with their local department of transportation to optimize network bandwidth and address network lag problems. In addition, from our experiments, GCP cloud instances will restart once a week for the purpose of host maintenance, thus a proper transition plan is needed for long-term operations. In addition, model fine-tuning is required to improve performance for scenarios like distant viewing angles, nighttime and rainy days.

Future work will pursue (1) more images from different viewing angles to fine-tune the current vehicle detection model, so as to improve our framework's performance in various environments, (2) a real-time alert system for deployment to help traffic operation centers address anomalous events quickly, (3) calibration of the image coordinates

of detected vehicles with the real-world geometric coordinates of the roadway network for more precise travel speed estimation and data visualization, and (4) the creation of modules using machine learning libraries for prediction applications such as travel speed estimation, congestion prediction.

**Acknowledgements** Our research results are based upon work supported by the Iowa DOT Office of Traffic Operations Support Grant. Any opinions, findings, conclusions or recommendations expressed in this material are that of the author(s) and do not necessarily reflect the views of the Iowa DOT Office of Traffic Operations.

## References

Bochinski E, Senst T, Sikora T (2018) Extending iou based multi-object tracking by visual information. In: 2018 15th IEEE international conference on advanced video and signal based surveillance (AVSS). IEEE, pp 1–6

Carol McDonald (2018) Real-time analysis of popular uber locations using apache APIs: Spark structured streaming, machine learning, Kafka and MapR database. https://mapr.com/blog/real-time-analysis-popular-uber-locations-spark-structured-streaming-machine-learning-kafka-and-mapr-db/. Accessed 10 Apr 2020

Chakraborty P, Adu-Gyamfi YO, Poddar S, Ahsani V, Sharma A, Sarkar S (2018) Traffic congestion detection from camera images using deep convolution neural networks. Transp Res Record J Transp Res Board 2672(45):222–231. https://doi.org/10.1177/0361198118777631

Dai Z, Song H, Wang X, Fang Y, Yun X, Zhang Z, Li H (2019) Video-based vehicle counting framework. IEEE Access 7:64460–64470

Docker (2020) Docker 19.03. https://www.docker.com/. Accessed 10 Mar 2020

Elastic (2020) elasticsearch-kibana 7.5.2. https://www.elastic.co/elasticsearch/. Accessed 21 Jan 2020

Eltoweissy M, Azab M, Olariu S, Gračanin D (2019) A new paradigm for a marketplace of services: smart communities in the IoT era. In: 2019 international conference on innovation and intelligence for informatics, computing, and technologies (3ICT). IEEE, pp 1–6

Ferdowsi A, Challita U, Saad W (2019) Deep learning for reliable mobile edge analytics in intelligent transportation systems: An overview. IEEE Veh Technol Mag 14(1):62–70

Gandy OH Jr, Nemorin S et al (2020) Transportation and smart city imaginaries: a critical analysis of proposals for the USDOT smart city challenge. Int J Commun 14:21

Girshick R (2015) Fast R-CNN. In: Proceedings of the IEEE international conference on computer vision, pp 1440–1448

Gstreamer (2019) Gstreamer open source multimedia framework. https://devblogs.nvidia.com/real-time-redaction-app-nvidia-deepstream-part-1-training/. Accessed 18 Dec 2019

He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 770–778

He K, Gkioxari G, Dollár P, Girshick R (2017) Mask R-CNN. In: Proceedings of the IEEE international conference on computer vision, pp 2961–2969

Iqbal K, Adnan M, Abbas S, Hasan Z, Fatima A (2018) Intelligent Transportation System (ITS) for smart-cities using mamdani fuzzy inference system. Int J Adv Comput Sci Appl. https://doi.org/10.14569/ijacsa.2018.090215

Kafka (2019) Kafka 2.4. https://kafka.apache.org/. Accessed 16 Dec 2019

Kumaran SK, Dogra DP, Roy PP (2019) Anomaly detection in road traffic using visual surveillance: a survey. arXiv preprint arXiv:190108292

Lin TY, Goyal P, Girshick R, He K, Dollár P (2017) Focal loss for dense object detection. In: Proceedings of the IEEE international conference on computer vision, pp 2980–2988

Liu H, Chen S, Kubota N (2013) Intelligent video systems and analytics: a survey. IEEE Trans Ind Inform 9(3):1222–1233

Liu W, Anguelov D, Erhan D, Szegedy C, Reed S, Fu CY, Berg AC (2016) Ssd: Single shot multibox detector. In: European conference on computer vision. Springer, pp 21–37

Liu P, Qi B, Banerjee S (2018) Edgeeye: An edge service framework for real-time intelligent video analytics. In: Proceedings of the 1st international workshop on edge systems, analytics and networking, pp 1–6

Lucas BD, Kanade T, et al. (1981) An iterative image registration technique with an application to stereo vision. In: Proceedings DAPPA image understanding workshop

Lukezic A, Vojir T, Cehovin Zajc L, Matas J, Kristan M (2017) Discriminative correlation filter with channel and spatial reliability. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 6309–6318

Memos VA, Psannis KE, Ishibashi Y, Kim BG, Gupta BB (2018) An efficient algorithm for media-based surveillance system (EAMSuS) in IoT smart city framework. Future Gener Comput Syst 83:619–628

Meng Q, Song H, Zhang Y, Zhang X, Li G, Yang Y (2020) Video-based vehicle counting for expressway: a novel approach based on vehicle detection and correlation-matched tracking using image data from PTZ cameras. Math Probl Eng 2020:1969408. https://doi.org/10.1155/2020/1969408

Mishra S, Patel S, Panda ARR, Mishra BK (2019) Exploring IoT-enabled smart transportation system. In: The IoT and the next revolutions automating the world. IGI Global, pp 186–202

NVIDIA Deepstream (2020) Deepstream 5.0. https://developer.nvidia.com/deepstream-sdk. Accessed 28 Apr 2020

NVIDIA IoT (2019) Smart parking detection. https://ngc.nvidia.com/catalog/containers/nvidia:deepstream$_$360d/. Accessed 27 Nov 2019

NVIDIA Metropolis (2019) Nvidia metropolis. https://mapr.com/blog/real-time-analysis-popular-uber-locations-spark-structured-streaming-machine-learning-kafka-and-mapr-db/0, accessed May 28, 2020

NVIDIA TLT (2020) Nvidia transfer learning toolkit. https://mapr.com/blog/real-time-analysis-popular-uber-locations-spark-structured-streaming-machine-learning-kafka-and-mapr-db/1. Accessed 10 Mar 2020

Petrolo R, Loscri V, Mitton N (2017) Towards a smart city based on cloud of things, a survey on the smart city vision and paradigms. Trans Emerg Telecommun Technol 28(1):e2931

Redmon J, Farhadi A (2018) Yolov3: an incremental improvement. arXiv preprint https://mapr.com/blog/real-time-analysis-popular-uber-locations-spark-structured-streaming-machine-learning-kafka-and-mapr-db/2

Ren S, He K, Girshick R, Sun J (2015) Faster R-CNN: towards real-time object detection with region proposal networks. In: Advances in neural information processing systems, pp 91–99

Shah et al (2020) Building a real-time redaction app using NVIDIA deepstream. https://devblogs.nvidia.com/real-time-redaction-app-nvidia-deepstream-part-1-training/. Accessed 12 Feb 2020

Spark (2020) Spark 2.4.5. https://mapr.com/blog/real-time-analysis-popular-uber-locations-spark-structured-streaming-machine-learning-kafka-and-mapr-db/4. Accessed 8 Feb 2020

Wojke N, Bewley A, Paulus D (2017) Simple online and realtime tracking with a deep association metric. In: 2017 IEEE international conference on image processing (ICIP). IEEE, pp 3645–3649. https://doi.org/10.1109/ICIP.2017.8296962